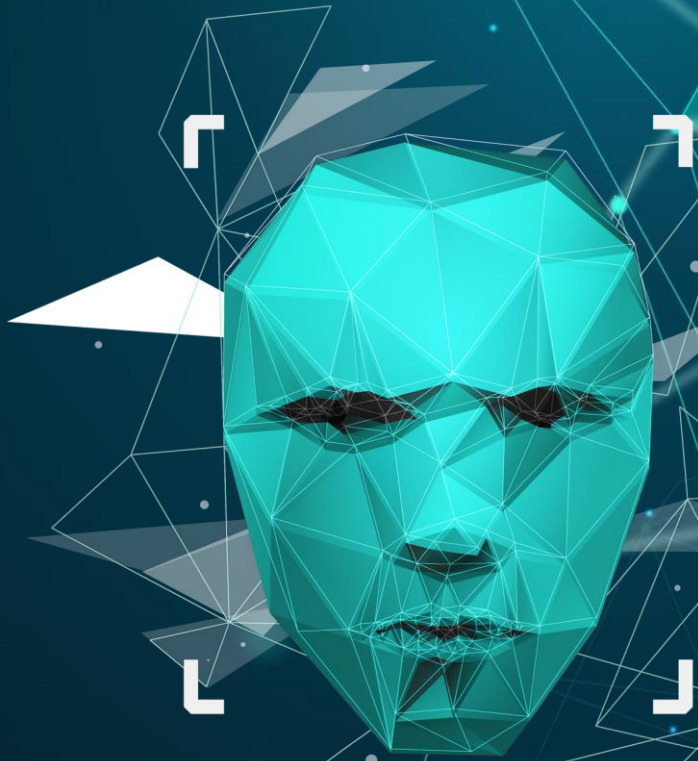
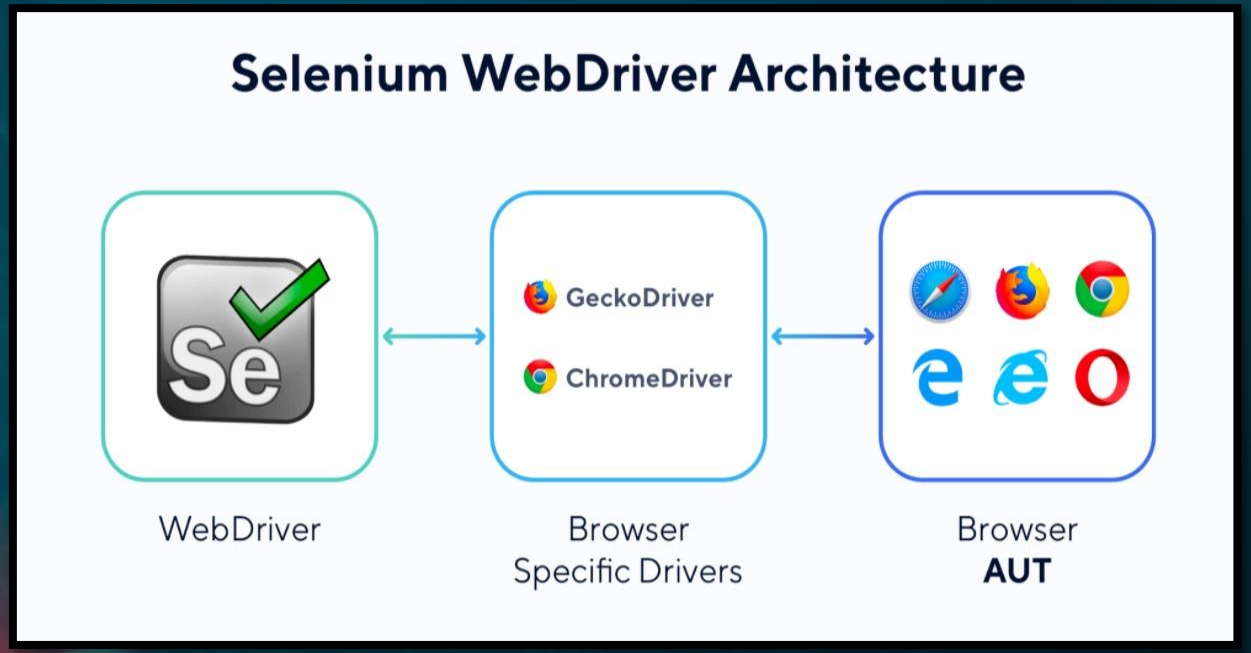
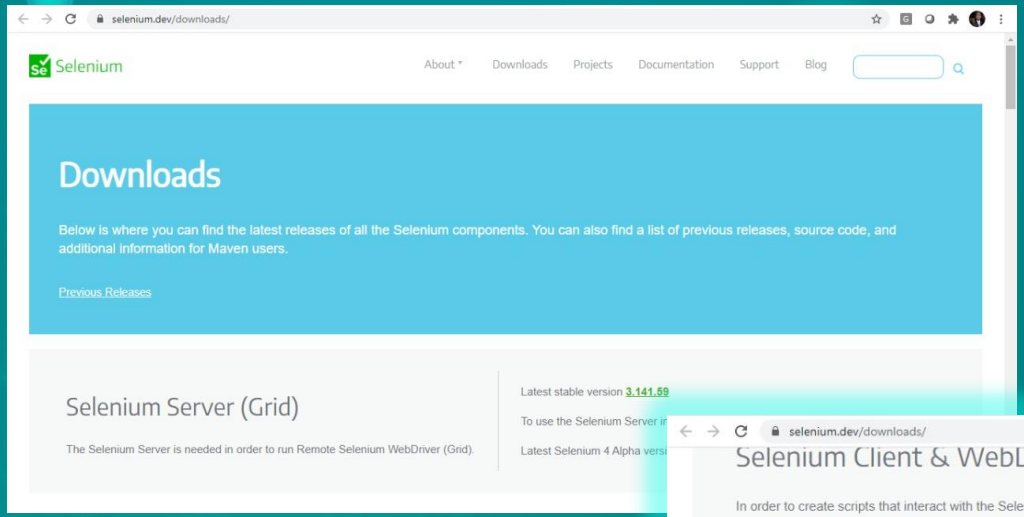
Selenium

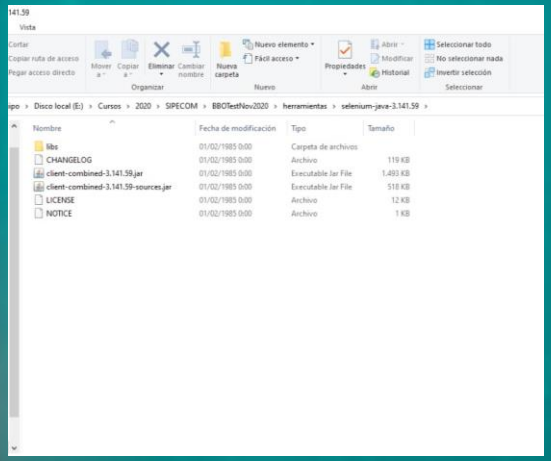


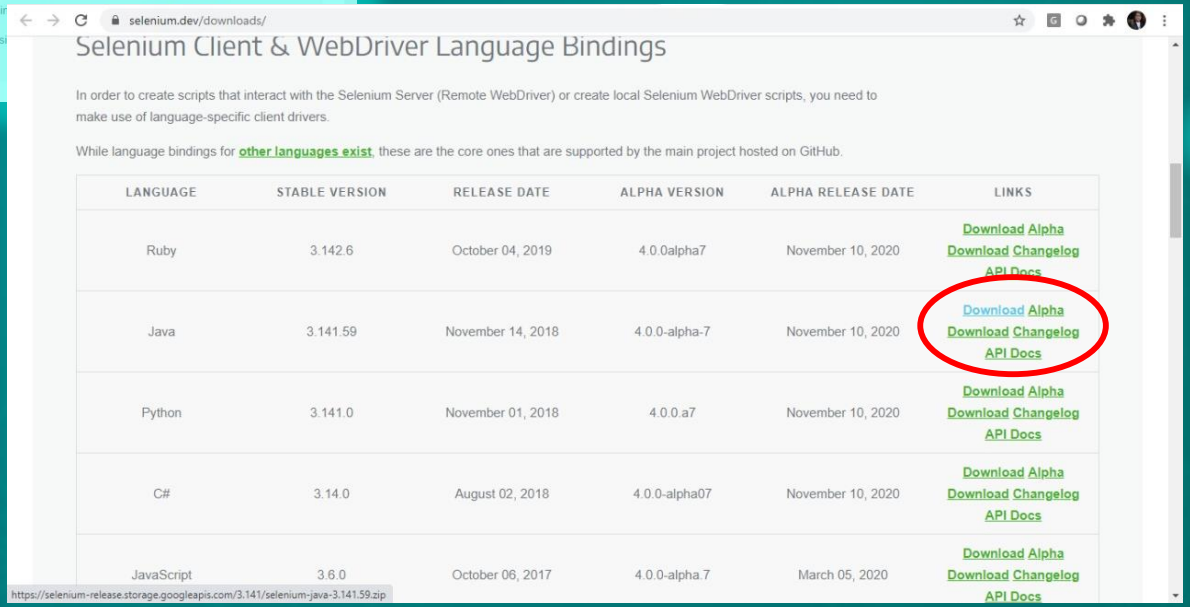
Victor Portugal.

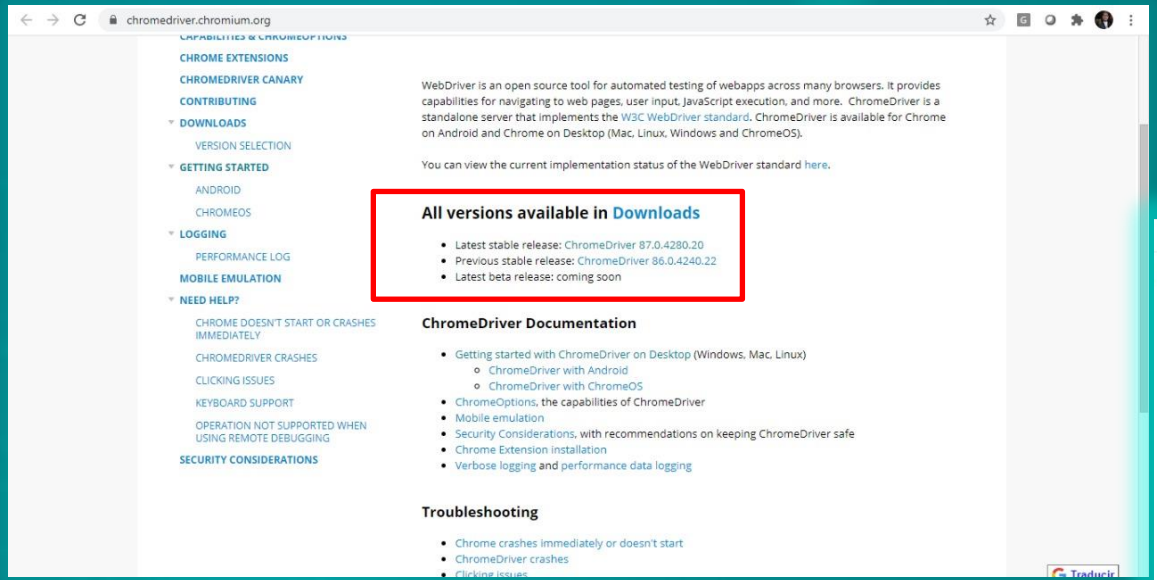

Selenium WebDriver Architecture
GeckoDriver
ChromeDriver
WebDriver
Browser
Specific Drivers
Browser
AUT

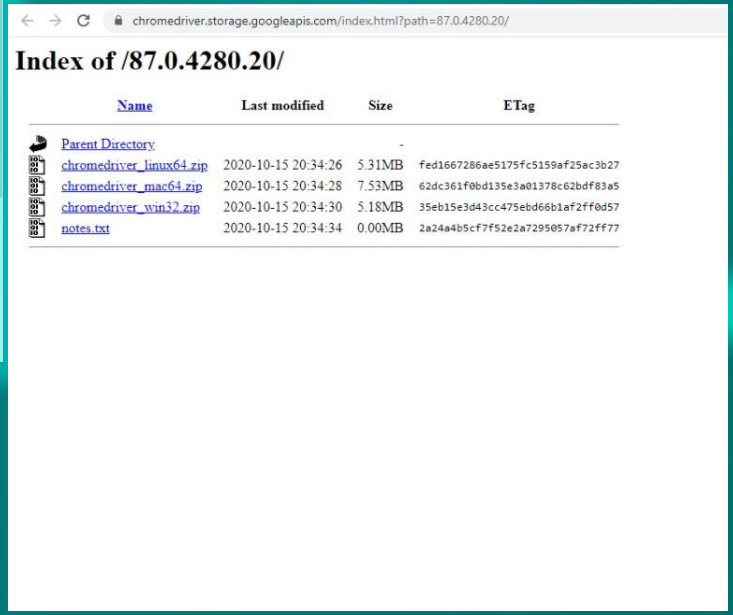
WebDriver

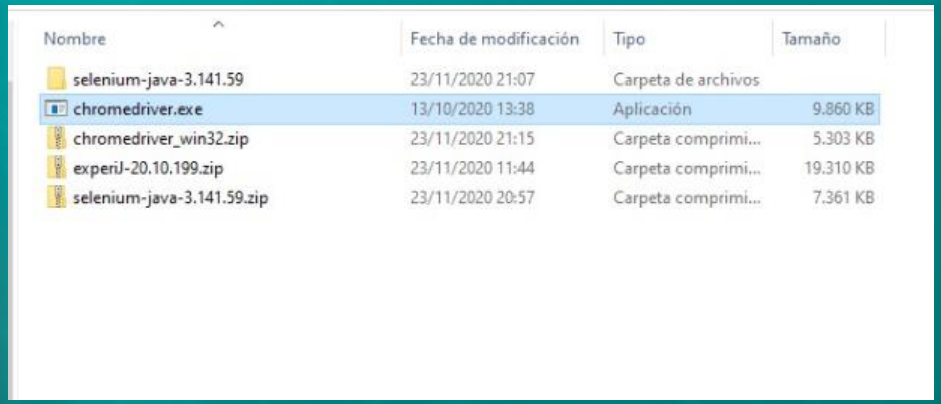

selenium.dev/downloads/
Selenium
About
Downloads
Projects
Documentation
Support
Blog
Downloads
Below is where you can find the latest releases of all the Selenium components. You can also find a list of previous releases, source code, and additional information for Maven users.
Previous Releases
Selenium Server (Grid)
The Selenium Server is needed in order to run Remote Selenium WebDriver (Grid).
Latest stable version 3.141.59

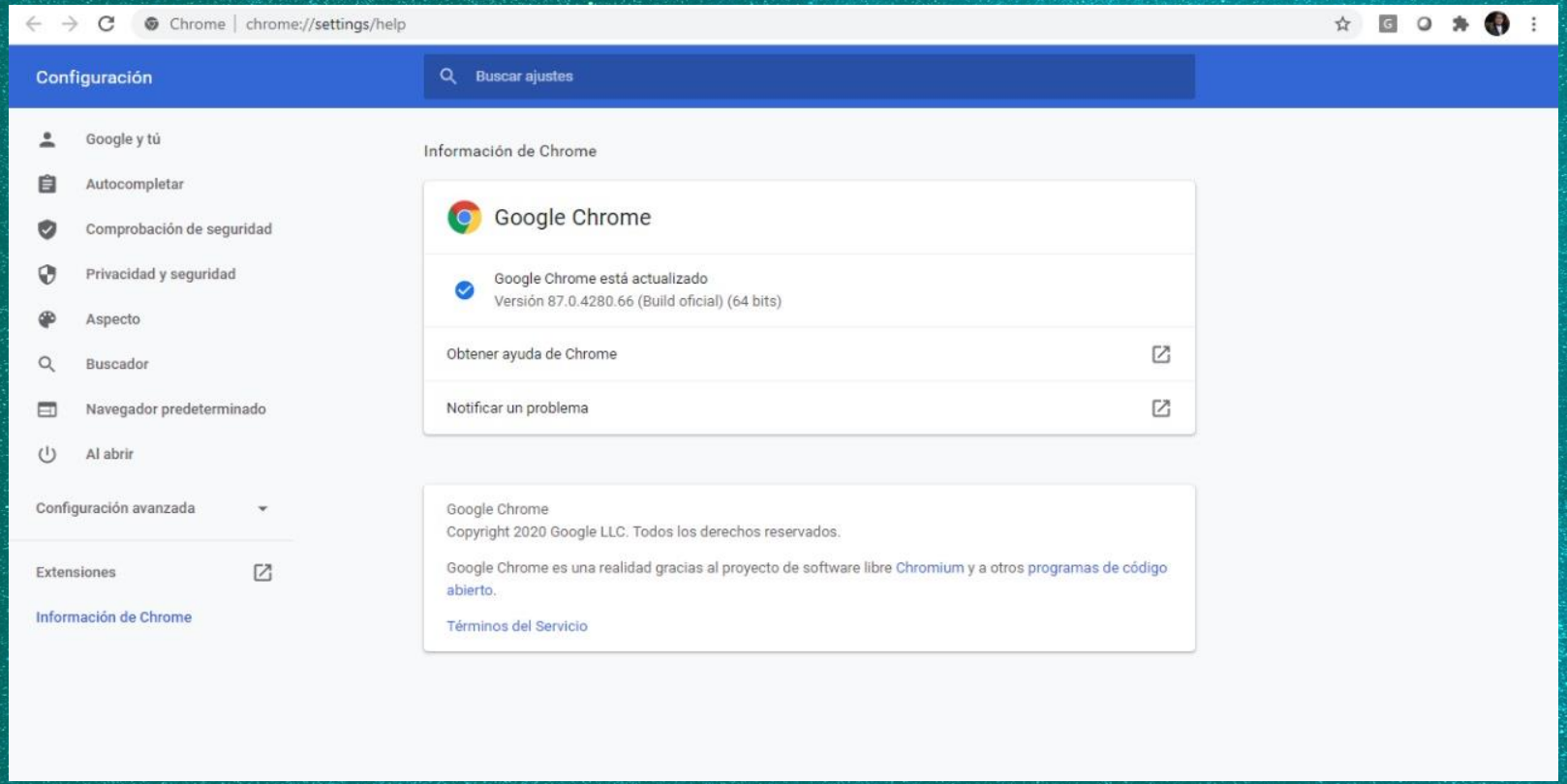

141.59
Vista
Disco local (E:) › Cursos › 2020 › SIPECOM › BBOTestNov2020 › herramientas › selenium-java-3.141.59 ›
Nombre
Fecha de modificación
Tipo
Tamaño
libs
CHANGELOG
client-combined-3.141.59.jar
client-combined-3.141.59-sources.jar
LICENSE
NOTICE

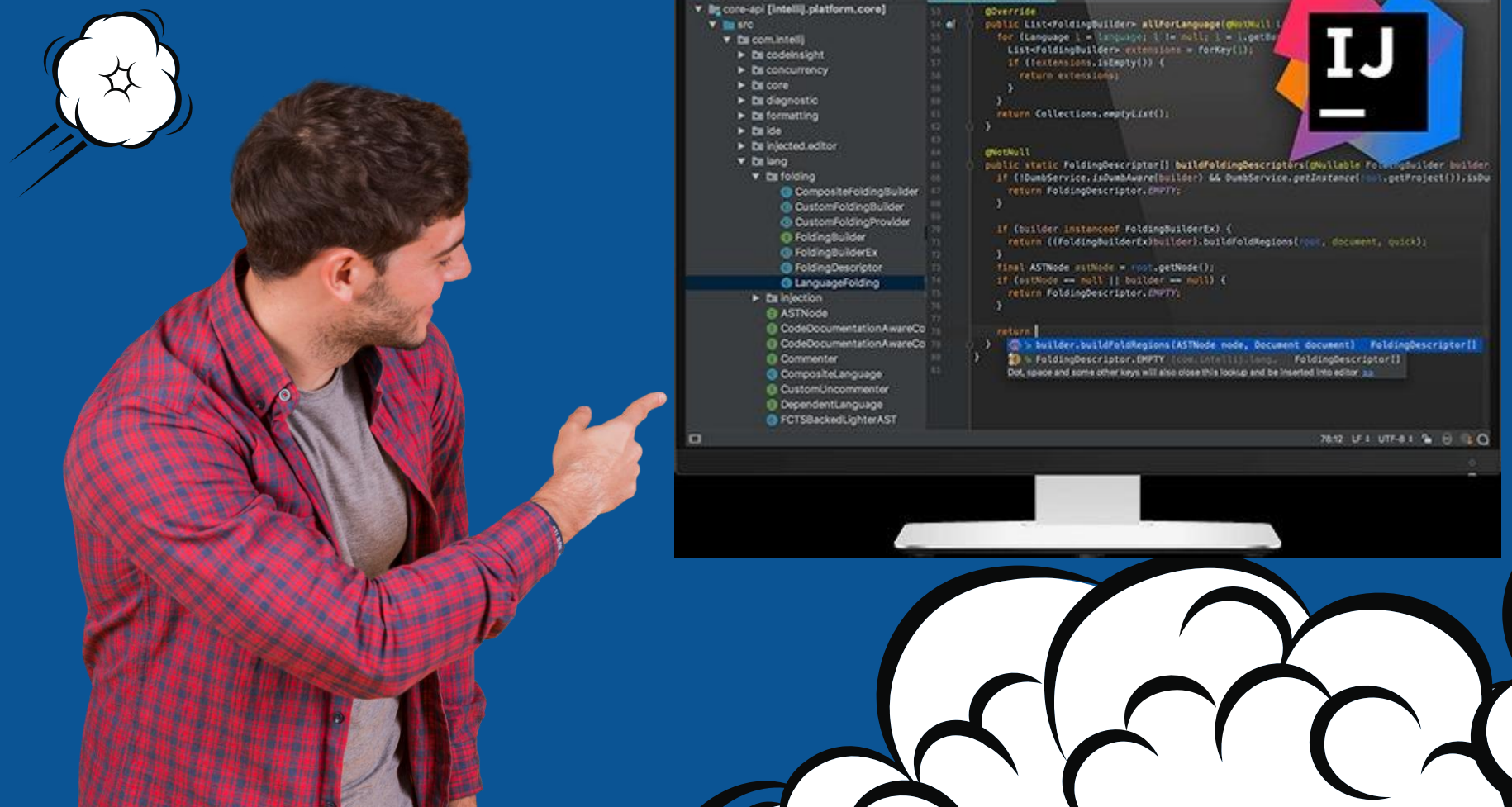

selenium.dev/downloads/
Selenium Client & WebDriver Language Bindings
In order to create scripts that interact with the Selenium Server (Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers.
While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on GitHub.
LANGUAGE | STABLE VERSION | RELEASE DATE | ALPHA VERSION | ALPHA RELEASE DATE | LINKS
Ruby | 3.142.6 | October 04, 2019 | 4.0.0alpha7 | November 10, 2020 | Download Alpha Download Changelog API Docs
Java | 3.141.59 | November 14, 2018 | 4.0.0-alpha-7 | November 10, 2020 | Download Alpha Download Changelog API Docs
Python | 3.141.0 | November 01, 2018 | 4.0.0.a7 | November 10, 2020 | Download Alpha Download Changelog API Docs
C# | 3.14.0 | August 02, 2018 | 4.0.0-alpha07 | November 10, 2020 | Download Alpha Download Changelog API Docs
JavaScript | 3.6.0 | October 06, 2017 | 4.0.0-alpha.7 | March 05, 2020 | Download Alpha Download Changelog API Docs
https://selenium-release.storage.googleapis.com/3.141/selenium-java-3.141.59.zip


chromedriver.chromium.org
CHROME EXTENSIONS
CHROMEDRIVER CANARY
CONTRIBUTING
DOWNLOADS
VERSION SELECTION
GETTING STARTED
ANDROID
CHROMEOS
LOGGING
PERFORMANCE LOG
MOBILE EMULATION
NEED HELP?
CHROME DOESN'T START OR CRASHES IMMEDIATELY
CHROMEDRIVER CRASHES
CLICKING ISSUES
KEYBOARD SUPPORT
OPERATION NOT SUPPORTED WHEN USING REMOTE DEBUGGING
SECURITY CONSIDERATIONS
WebDriver is an open source tool for automated testing of webapps across many browsers. It provides capabilities for navigating to web pages, user input, JavaScript execution, and more. ChromeDriver is a standalone server that implements the W3C WebDriver standard. ChromeDriver is available for Chrome on Android and Chrome on Desktop (Mac, Linux, Windows and ChromeOS).
You can view the current implementation status of the WebDriver standard here.
All versions available in Downloads
Latest stable release: ChromeDriver 87.0.4280.20
Previous stable release: ChromeDriver 86.0.4240.22
Latest beta release: coming soon
ChromeDriver Documentation
Getting started with ChromeDriver on Desktop (Windows, Mac, Linux)
ChromeDriver with Android
ChromeDriver with ChromeOS
ChromeOptions, the capabilities of ChromeDriver
Mobile emulation
Security Considerations, with recommendations on keeping ChromeDriver safe
Chrome Extension installation
Verbose logging and performance data logging
Troubleshooting
Chrome crashes immediately or doesn't start
ChromeDriver crashes
Clicking issues

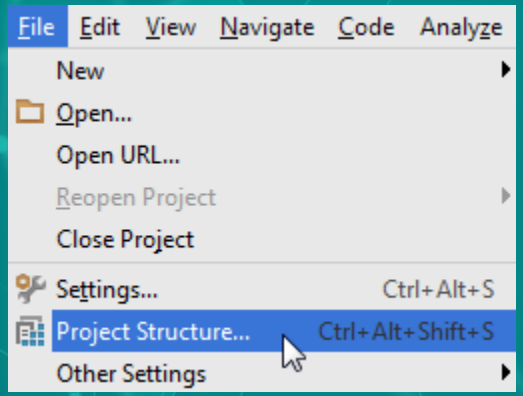

chromedriver.storage.googleapis.com/index.html?path=87.0.4280.20/
Index of /87.0.4280.20/
Name
Last modified
Size
ETag
Parent Directory -
chromedriver_linux64.zip 2020-10-15 20:34:26 5.31MB fed1667286ae5175fc5159af25ac3b27
chromedriver_mac64.zip 2020-10-15 20:34:28 7.53MB 62dc361f0bd135e3a01378c62bdf83a5
chromedriver_win32.zip 2020-10-15 20:34:30 5.18MB 35eb15e3d43cc475ebd66b1af2ff0d57
notes.txt 2020-10-15 20:34:34 0.00MB 2a24a4b5cf7f52e2a7295057af72ff77


Nombre
Fecha de modificación
Tipo
Tamaño
selenium-java-3.141.59 23/11/2020 21:07 Carpeta de archivos
chromedriver.exe 13/10/2020 13:38 Aplicación 9.860 KB
chromedriver_win32.zip 23/11/2020 21:15 Carpeta comprimi... 5.303 KB
experiJ-20.10.199.zip 23/11/2020 11:44 Carpeta comprimi... 19.310 KB
selenium-java-3.141.59.zip 23/11/2020 20:57 Carpeta comprimi... 7.361 KB

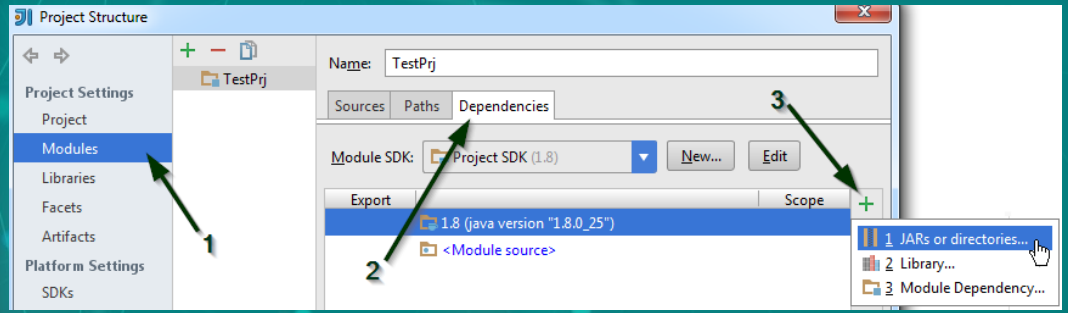

Chrome | chrome://settings/help
Configuración
Buscar ajustes
Google y tú
Autocompletar
Comprobación de seguridad
Privacidad y seguridad
Aspecto
Buscador
Navegador predeterminado
Al abrir
Configuración avanzada
Extensiones
Información de Chrome
Información de Chrome
Google Chrome
Google Chrome está actualizado
Versión 87.0.4280.66 (Build oficial) (64 bits)
Obtener ayuda de Chrome
Notificar un problema
Google Chrome
Copyright 2020 Google LLC. Todos los derechos reservados.
Google Chrome es una realidad gracias al proyecto de software libre Chromium y a otros programas de código abierto.
Términos del Servicio



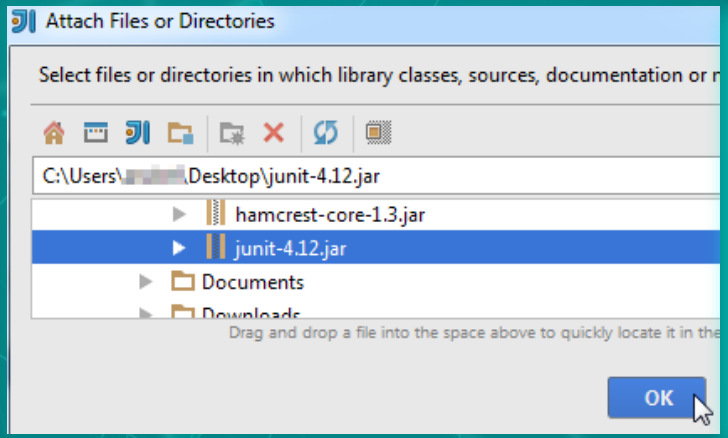
1.- File > **Project Structure...**

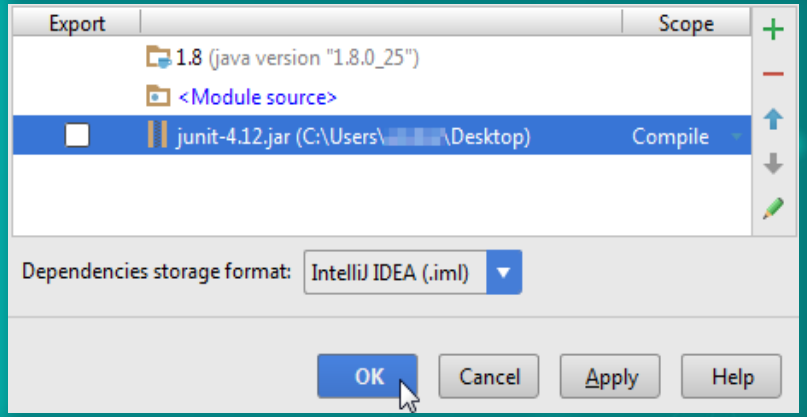


2- Project Settings > **Modules** > **Dependencies** > "**+**" sign > **JARs or directories...**

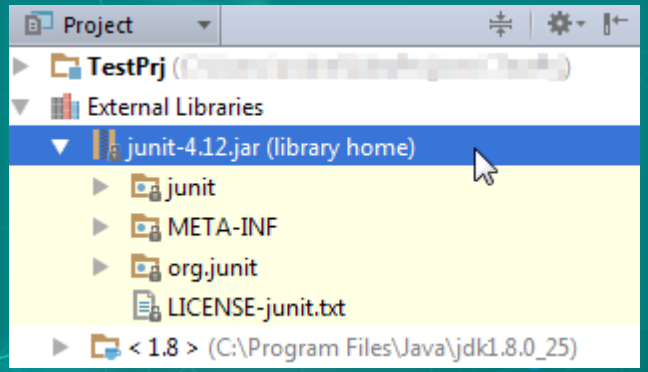


3.- Select the jar file and click on OK, then click on another OK button to confirm

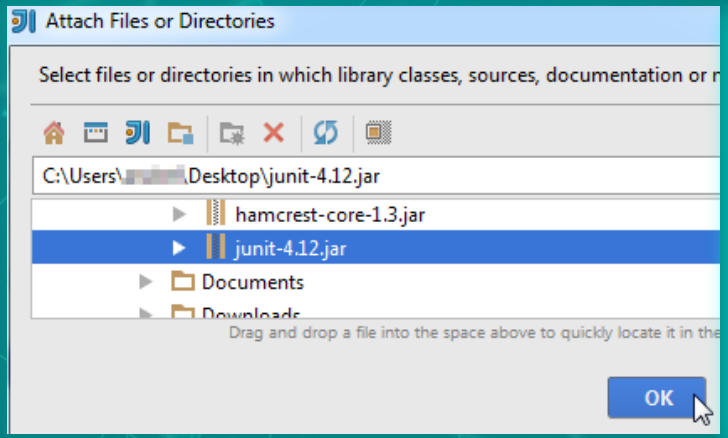


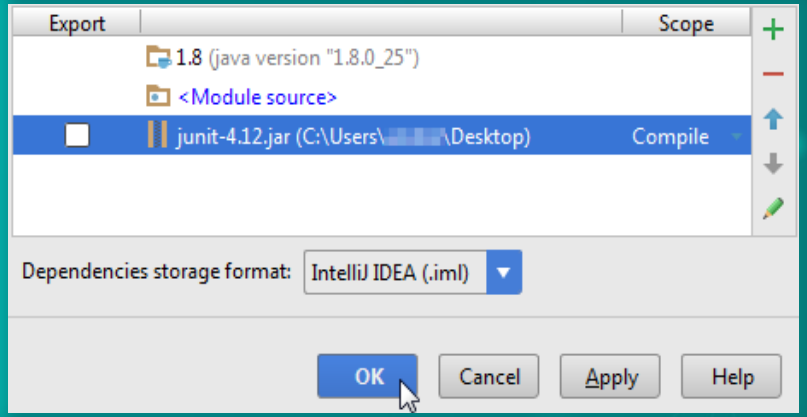


1. ## 4- You can view the jar file in the "External Libraries" folder

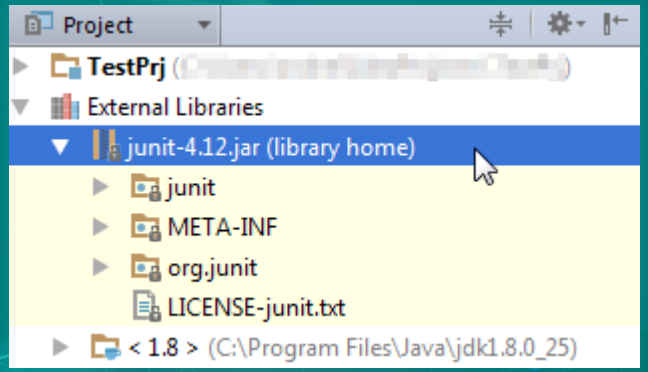


3.- Select the jar file and click on OK, then click on another OK button to confirm

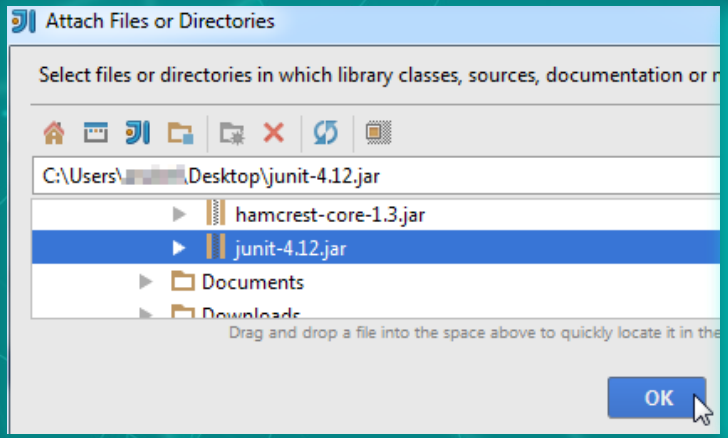


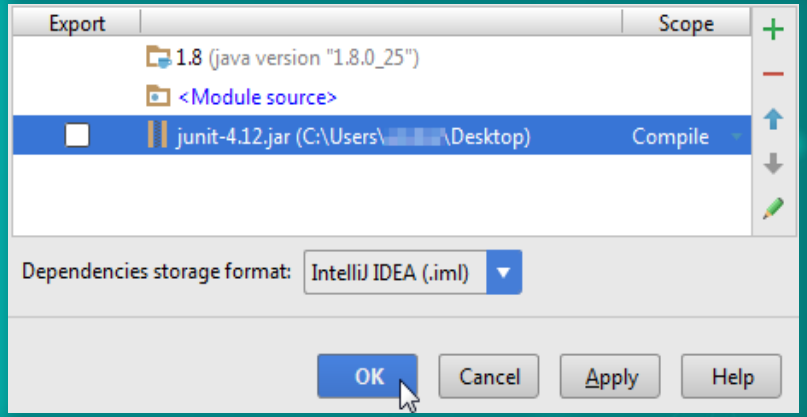


1. ## 4- You can view the jar file in the "External Libraries" folder

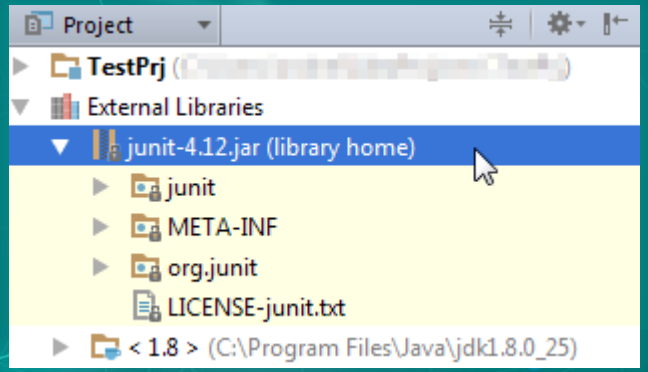


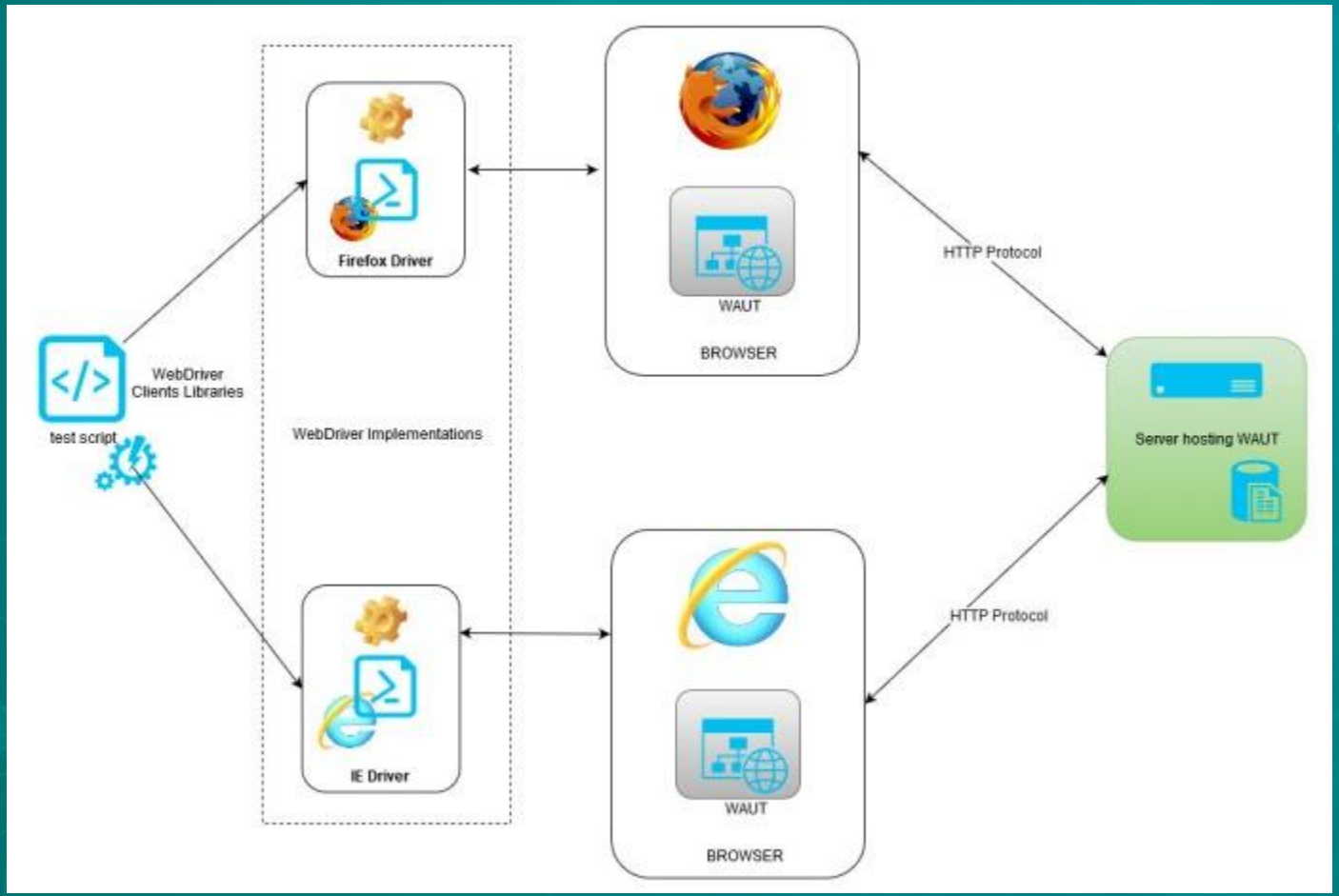
3.- Select the jar file and click on OK, then click on another OK button to confirm





1. ## 4- You can view the jar file in the "External Libraries" folder




Firefox Driver
WAUT
BROWSER
HTTP Protocol
WebDriver
Clients Libraries
test script
WebDriver Implementations
Server hosting WAUT
IE Driver
WAUT
BROWSER
HTTP Protocol

¿Cómo empezamos…?

MÉTODO get()

Return	Método(Parámetros)
void	get(java.lang.String url)

Descripción

- Método que se encarga de acceder a la url que se le pasa como parámetro y cargar su contenido en una ventana del navegador web.
- Si el navegador no se encuentra abierto, primero lo abre, y posteriormente carga la página web en una ventana nueva.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
driver.get("www.google.es");
```

MÉTODO getCurrentUrl()

Return	Método(Parámetros)
java.lang.String	getCurrentUrl()

Descripción

- Método que devuelve la url que está mostrando el navegador en el momento de su llamada.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
driver.get("www.google.es");
System.out.println(driver.getCurrentUrl());
```

MÉTODO getTitle()

Return	Método(Parámetros)
java.lang.String	getTitle()

Descripción

- Método que devuelve el título de la ventana que está mostrando el navegador en el momento de su llamada.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
driver.get("www.google.es");
System.out.println(driver.getTitle());
```

MÉTODO getPageSource()

Return	Método(Parámetros)
java.lang.String	getPageSource()

Descripción

- Método que devuelve el código fuente HTML de la página web que está mostrando el navegador en el momento de su llamada.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
driver.get("www.google.es");
System.out.println(driver.getPageSource());
```

MÉTODO close()

Return	Método(Parámetros)
void	Close()

Descripción

- Método que cierra la ventana activa que esté mostrando el navegador en el momento de su llamada. Si es la última ventana, también cierra el navegador.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
driver.get("www.google.es");
driver.close();
```

Ejemplo cap7.Ejemplo1.java

```
/*
 * EJEMPLO 1: Métodos básicos interfaz Webdriver
 * ACCIONES:
 * Cargar la pagina web
 * Mostrar por consola la url de la pagina cargada
 * Mostrar por consola el titulo de la ventana abierta
 * Mostrar por consola el codigo HTML de la pagina cargada
*/

WebDriver driver = new FirefoxDriver();
driver.get("file:///C:/WebSites/localhost/mantisbt-1.2.19/login_page.htm");

System.out.println(driver.getCurrentUrl());
System.out.println(driver.getTitle());
System.out.println(driver.getPageSource());

Thread.sleep(5000);
driver.close();
```

Después de cargar una determinada página web, el siguiente paso es poder localizar los diferentes elementos que la componen con el objetivo de analizarlos o actuar sobre ellos. Por ejemplo, en una aplicación web donde para acceder a un determinado servicio se nos ofrece un portal de inicio de sesión, seguramente nos interese localizar los campos donde poder introducir el usuario y la contraseña.

Cada elemento que compone una página web Selenium WebDriver lo representa mediante un objeto de tipo WebElement. Los cajetines de texto anteriormente mencionados, donde poder introducir el usuario y la contraseña, serán manejados a partir de su representación como objetos de este tipo.

La interfaz WebDriver define 2 métodos para la localización de elementos:

MÉTODO findElement()

Return	Método(Parámetros)
WebElement	findElement(By by)

Descripción

- Método que devuelve el primer elemento encontrado que cumpla con el tipo de búsqueda (mecanismo) definido por el parámetro by.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
…
WebElement user = driver.findElement(By.name("username"));
```

Ejemplo cap7.Ejemplo2.java

```
/*
 * EJEMPLO 2: Localización de WebElements
 * ACCIONES:
 * Cargar la página web
 * Localizar cajetin para introducir usuario
 * Teclear texto
 * Borrar texto
 * Teclear texto
*/

WebDriver driver = new FirefoxDriver();
driver.get("file:///C:/WebSites/localhost/mantisbt-1.2.19/login_page.htm");

//Localizar cajetin para introducir el usuario
WebElement user = driver.findElement(By.name("username"));
Thread.sleep(1000);
//Escribir sobre el cajetin
user.sendKeys("Usuario");

Thread.sleep(2000);
//Borrar el contenido del cajetin
user.clear();
Thread.sleep(500);
//Volver a escribir en el cajetin
user.sendKeys("OtroUsuario");

Thread.sleep(2000);
driver.close();
```

```
/*
 * EJEMPLO 4: Localización de WebElements. Ejemplo 8 Mecanismos By
 * ACCIONES:
 * Localizar elemento ByClassName y añadir a la lista
 * Localizar elemento ByName y añadir a la lista
 * Localizar elemento ByTagName y añadir a la lista
 * Localizar elemento ById y añadir a la lista
 * Localizar elemento ByLinkText y añadir a la lista
 * Localizar elemento ByPartialLinkText y añadir a la lista
 * Localizar elemento ByXpath y añadir a la lista
 * Localizar elemento ByCssSelector y añadir a la lista
 * Recorrer la lista de elementos imprimiendo informacion de cada uno de
 * ellos
*/

WebDriver driver = new FirefoxDriver();
driver.get("file:///C:/WebSites/localhost/mantisbt-1.2.19/login_page.htm");

List<WebElement> listaElementos = new ArrayList<WebElement>();
//Localizar elemento ByClassName
listaElementos.add(driver.findElement(By.className("button")));

//Localizar elemento ByName
listaElementos.add(driver.findElement(By.name("username")));

//Localizar elemento ByTagName
listaElementos.add(driver.findElement(By.tagName("td")));

//Localizar elemento ById
listaElementos.add(driver.findElement(By.id("session")));

//Localizar elemento ByLinkText
listaElementos.add(driver.findElement(By.linkText("Signup for a new
account")));

//Localizar elemento ByPartialLinkText
listaElementos.add(driver.findElement(By.partialLinkText("Lost")));

//Localizar elemento ByXpath
listaElementos.add(driver.findElement(By.xpath("html/body/div[3]/form/table/t
body/tr[4]/td[1]")));

//Localizar elemento ByCssSelector
listaElementos.add(driver.findElement(By.cssSelector("body > table")));

Iterator<WebElement> it = listaElementos.iterator();
WebElement e = null;
while(it.hasNext()){
      e = it.next();
      System.out.println(e.toString());
      System.out.println("Tag: " + e.getTagName() + "  Position: " +
      e.getLocation().toString());
      System.out.println("Text: " + e.getText()+ "\n");
}
Thread.sleep(2000);
driver.close();
```

2015

MÉTODO WebElement.sendKeys ()

La acción de poder introducir texto en un elemento de tipo “caja de texto” o de tipo “área de texto” se puede realizar mediante el método sendKeys().

Return	Método(Parámetros)
void	sendKeys(java.lang.String keysToSend)

Descripción

```
- Método que simula la introducción de texto por parte del ususario en
elementos de tipo caja de texto o área de texto.
- Si el elemento en cuestión no soporta la introducción de texto, el método
no produce acción alguna.

Ejemplo:

WebDriver driver = new FirefoxDriver();
…
WebElement user = driver.findElement(By.name("username"));
user.sendKeys("Miguel");

NOTA: Para simular la pulsación de ciertas teclas especiales es necesario
ayudarse de la clase Keys que proporciona las funcionalidades necesarias para
emular el teclado de un PC. Para más información, consultar el recurso web
http://seleniumhq.github.io/selenium/docs/api/java/

Ejemplo:

user.sendKeys(Keys.chord(Keys.SHIFT, "Miguel"));
```

MÉTODO WebElement.clear()

Si con el anterior método, sendKeys(), se podía simular la introducción de texto sobre un elemento web, con el método clear() se puede simular la acción de borrarlo.

Return	Método(Parámetros)
void	clear()

Descripción

- Método que borra el texto introducido en elementos de tipo caja de texto o área de texto.
- Si el elemento en cuestión no soporta la introducción/borrado de texto, el método no produce acción alguna.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
…
WebElement user = driver.findElement(By.name("username"));
user.sendKeys("Miguel");
user.clear()
```

MÉTODO WebElement.submit()

Cuando el elemento web en cuestión es un formulario, o está dentro de un formulario, el método submit() valida el formulario y lo envía al servidor.

Return	Método(Parámetros)
void	submit()

Descripción

- Método para validar y enviar un elemento de tipo formulario o perteneciente a un formulario.
- Si el elemento seleccionado no es o forma parte de un formulario, el método submit() lanza una excepción NoSuchElementException.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
…
WebElement user = driver.findElement(By.name("username"));
user.sendKeys("Miguel");
user.submit()
```

MÉTODO WebElement.getText()

En ocasiones, nos interesará comprobar el texto que está mostrando un determinado elemento web. Para ello WebElement ofrece el método getText()

Return	Método(Parámetros)
java.lang.String	getText()

Descripción

- Método para obtener el texto mostrado o contenido por un elemento.
- Devuelve una cadena de caracteres. Si el elemento no contiene texto, la cadena devuelta será vacía.
- Método aplicable a cualquier elemento web.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
…
WebElement e = driver.findElement(By.tagName("address"));
System.out.println(e.getText());
```

MÉTODO WebElement.isEnable()

Para conocer si un determinado elemento está habilitado o no, utilizaremos el método isEnable().

Return	Método(Parámetros)
boolean	isEnable()

Descripción

- Método para conocer si un elemento esta habilitado. Devuelve True o False en función de si lo está o no.
- Método aplicable a cualquier elemento web.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
…
WebElement button = driver.findElement(By.className("button"));
System.out.println(button.isEnable())
```

MÉTODO WebElement.click()

Cuando necesitemos hacer “clic” sobre un determinado elemento utilizaremos el método click().

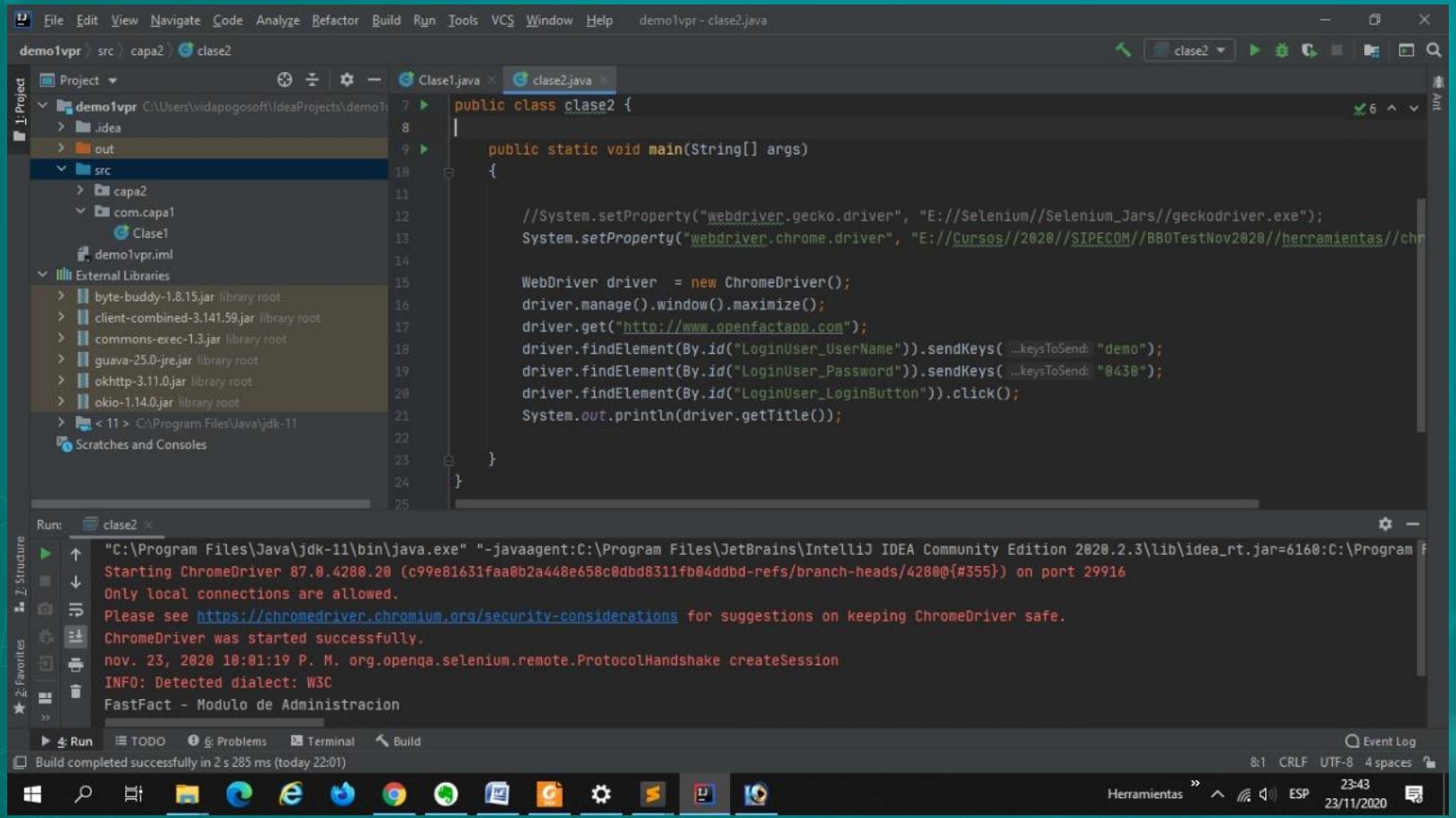
Return	Método(Parámetros)
void	click()

Descripción

- Método para realizar clic sobre el elemento web seleccionado.
- Hay que tener en cuenta que si ésta acción tienen como consecuencia un cambio de página, todas las referencias a elementos web pueden cambiar.

Ejemplo:

```
WebDriver driver = new FirefoxDriver();
…
WebElement button = driver.findElement(By.className("button"));
button.click()
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help demo1vpr - clase2.java
demo1vpr src capa2 clase2
Project
demo1vpr C:\Users\vidapogosoft\IdeaProjects\demo1
.idea
out
src
capa2
com.capa1
Clase1
demo1vpr.iml
External Libraries
byte-buddy-1.8.15.jar library root
client-combined-3.141.59.jar library root
commons-exec-1.3.jar library root
guava-25.0-jre.jar library root
okhttp-3.11.0.jar library root
okio-1.14.0.jar library root
< 11 > C:\Program Files\Java\jdk-11
Scratches and Consoles
Clase1.java
clase2.java
public class clase2 {
public static void main(String[] args)
{
//System.setProperty("webdriver.gecko.driver", "E://Selenium//Selenium_Jars//geckodriver.exe");
System.setProperty("webdriver.chrome.driver", "E://Cursos//2020//SIPECOM//BBOTestNov2020//herramientas//chr
WebDriver driver = new ChromeDriver();
driver.manage().window().maximize();
driver.get("http://www.openfactapp.com");
driver.findElement(By.id("LoginUser_UserName")).sendKeys(...keysToSend: "demo");
driver.findElement(By.id("LoginUser_Password")).sendKeys(...keysToSend: "0430");
driver.findElement(By.id("LoginUser_LoginButton")).click();
System.out.println(driver.getTitle());
}
}
Run: clase2
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.3\lib\idea_rt.jar=6160:C:\Program
Starting ChromeDriver 87.0.4280.20 (c99e81631faa0b2a448e658c0dbd8311fb04ddbd-refs/branch-heads/4280@{#355}) on port 29916
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
nov. 23, 2020 10:01:19 P. M. org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
FastFact - Modulo de Administracion
4: Run TODO 6: Problems Terminal Build Event Log
Build completed successfully in 2 s 285 ms (today 22:01)
8:1 CRLF UTF-8 4 spaces
Herramientas ESP 23:43 23/11/2020